UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/608,040	06/30/2003	Hajime Ogawa	2003_0866A	3923

513 7590 03/26/2007

WENDEROTH, LIND & PONACK, L.L.P.
2033 K STREET N. W.
SUITE 800
WASHINGTON, DC 20006-1021

EXAMINER	
WEI, ZHENG	
ART UNIT	**PAPER NUMBER**
2192	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	03/26/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)	
	10/608,040	OGAWA ET AL.	
	Examiner	Art Unit	
	Zheng Wei	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *30 June 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-43* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-43* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *30 June 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☒ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *01/24/06, 10/04/06*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This office action is in response to the application filed on 06/30/2003.

2. Claims 1-43 are pending and have been examined.

Oath/Declaration

3. The Office acknowledges receipt of a properly signed oath/declaration filed on June 30, 2003.

Priority

4. The priority date considered for this application is July 03, 2002, which is the filing date of the Application No. JP 2002-195305. A certified copy of the priority application has been received and placed in the application file.

Information Disclosure Statement

5. The information disclosure statements filed 01/24/2006 and 10/04/2006 have been placed in the application file and the information referred to therein has been considered.

Drawings

6. The drawings filed on June 30, 2003 are accepted by the Examiner.

Specification

7. The abstract of the disclosure is objected to because the abstract should be in narrative form and generally limited to **a single paragraph within the range of 50 to 150 words.** Correction is required. See MPEP § 608.01(b).

Claim Objections

8. Claims 3 and 4 are objected to because of the following informalities:

Claims 3 and 4: "the optimization unit allocates array data that are an object of a directive" should be changed to --the optimization unit allocates array data that **is** an object of a directive--

Appropriate correction is required.

Claim Rejections - 35 USC § 112

9. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

10. Claim 2, 30 and 34 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 2: The term "out of array data" in claim 2 is a relative term which renders the claim indefinite. The term "out of array data" is not defined by the claim, the specification does not provide a standard for ascertaining the requisite degree, and one of ordinary skill in the art would not be reasonably apprised of the scope of the invention. For the purpose of compact prosecution, the Examiner does not consider the sentence "out of array data declared by the source program" as part of claim.

Claims 30 and 34: The term "special type" in claims 30 and 34 is a relative term which renders the claim indefinite. The term "special type" is not defined by the claim, the specification does not provide a standard for ascertaining the requisite degree, and one of ordinary skill in the art would not be reasonably apprised of the scope of the invention. For the purpose of compact prosecution, the Examiner treats the "special type" as –data type--.

Claim Rejections - 35 USC § 101

11. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

12. Claims 1-43 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claims 1, 6, 14, 24, 29: These claims merely claim compiler apparatuses that comprise a directive acquisition unit and an optimization unit which are only software modules and thus can be interpreted as software compiler program listings per se. The descriptions or expressions of the programs are not physical "things". They are neither computer components nor statutory processes, as they are not "acts" being performed. Such claimed computer programs do not define any structural and functional interrelationships between the computer software and hardware components which permit the computer program's functionality to be realized. Therefore, computer programs (compilers) claimed as computer listings per se are nonstatutory. See M.P.E.P. 2106.01 (I)

Claims 2-4, 7-12, 15-22, 25-27 and 30-34, 41-43: These claims, which depend from claims 1, 6, 14, 24 and 29 respectively, do not remedy the deficiencies as noted above, thus are also rejected under 35 U.S.C. 101 for the same reasons.

Claims 5, 13, 23, 28 and 35: These claims claim the source programs, which are recorded on computer-readable recording medium, wherein the source program includes at least one of descriptions. However, source programs including descriptions themselves are "Nonfunctional descriptive material" and when nonfunctional descriptive materials are recorded on some computer-readable medium, in a computer or on electromagnetic carrier signals, they are not

statutory since no requisite functionalities are present to satisfy the practical, useful application requirements. See M.P.E.P. 2106.01 (II)

Claims 36-40: These claims claim programs for compiler apparatuses. However, both "program" and "compiler apparatus" are computer listings per se. They are neither computer components nor statutory processes, as they are not "acts" being performed. Such claimed computer programs do not define any structural and functional interrelationships between the computer program and other claimed elements of a computer (hardware), which permit the computer programs functionality to be realized. Therefore, computer programs and compiler apparatuses claimed as computer listings per se are nonstatutory. See M.P.E.P. 2106.01 (I)

Claim Rejections - 35 USC § 102

13. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

14. Claims 14-17, 29-34 and 41-43 are rejected under 35 U.S.C. 102(b) as being anticipated by Stallman (Richard M. Stallman, Using and Porting the GNU Compiler Collection for GCC 3.1)

Claim 14:

Stallman discloses a compiler apparatus that translates a source program into a machine language program comprising:

- a directive acquisition unit operable to acquire a directive for optimizing a machine language program to be generated (see for example, p.9, "Optimization options"); and
- an optimization unit operable to perform optimization by generating a sequence of machine language instructions following an acquired directive (see for example, see p.49, section 3.10, "Options that Control Optimization"),

wherein the optimization unit performs optimization by loop unrolling following a directive when the directive acquisition unit acquires the directive on the optimization by loop unrolling (see for example, p.9-10, section optimization option, "-funroll-loops, -funroll-all-loops")

Claim 15:

Stallman further discloses the compiler apparatus according to claim 14,

- wherein the directive acquisition unit acquires a directive for not performing the optimization by loop unrolling together with a directive for translating the

source program (see for example, p.49, section 3.10 Option That Control Optimization, option "-O" and related text description, also see p.50, lines 29-32, "positive and negative forms, the negative form of '-ffoo' would be '-fno-foo'". Therefore, the negative form of '-funroll-loops" is '-fnounroll-loops' that can be used to direct compiler not performing loop unrolling) and

- the optimization unit restrains the optimization by loop unrolling of all the loop processing in the source program when the directive acquisition unit acquires the directive for not performing the optimization by loop unrolling (see for example, see p.49, section 3.10, "Options that Control Optimization", and also see p.55, lines 38-42, "-funroll-loops" and related description).

Claim 16:

Stallman also discloses the compiler apparatus according to claim 14,

- wherein the directive acquisition unit detects a directive for performing the optimization by loop unrolling of a specific loop processing in the source program (see for example, p.55, lines 38-42, "This is only done for loops whose number of iterations can be determined at compile time or run time"), and
- the optimization unit performs the optimization by loop unrolling of loop processing that is an object of the directive detected by the directive

acquisition unit (see for example, p.55, lines 38-42, "Perform the optimization of loop unrolling").

Claim 17:

Stallman The compiler apparatus according to claim 14,

- wherein the directive acquisition unit detects a directive for not performing the optimization by loop unrolling of a specific loop processing in the source program (see for example, p.55, lines 38-42, "This is only done for loops whose number of iterations can be determined at compile time or run time". It also implies that this directive also directs not performing the optimization for those whose number of iterations can not be determined at compile time or run time), and

the optimization unit restrains the optimization by loop unrolling of loop processing that is an object of the directive detected by the directive acquisition unit (see for example, p.55, lines 38-42, "Perform the optimization of loop unrolling").

Claim 29:

Stallman discloses a compiler apparatus that translates a source program into a machine language program comprising:

- a directive acquisition unit operable to acquire a directive for optimizing a machine language program to be generated (see for example, p.9, "Optimization options"); and
- an optimization unit operable to perform optimization by generating a sequence of machine language instructions following the acquired directive (see for example, see p.49, section 3.10, "Options that Control Optimization"),

wherein the optimization unit performs optimization by allocating data in a memory region following a directive when the optimization unit acquires the directive on alignment of the array data to be allocated in a memory region (see for example, p.177, section 5.33 Specifying Attributes of Variables. "aligned (alignment)" and related description, also see example, p.178, line 7 and line 19, "short array[3] __attribute__ ((aligned)):" and related description)

Claim 30:

Stallman further discloses the compiler apparatus according to claim 29,

- wherein the directive acquisition unit acquires a directive for alignment of array data of a special type together with a directive for translating the source program (see for example, p.10, line 3, "-O –O0 –O1 –O2 -O3 –Os" and related text), and
- the optimization unit allocates all the array data of the special type declared in the source program in the memory region so that its head address

matches the alignment (see for example, p.177, section 5.33 Specifying Attributes of Variables. "aligned (alignment)" and related description, also see example, p.178, line 1-19, "short array[3] __attribute__ ((aligned)):" and related description).

Claim 34:

Stallman also discloses the compiler apparatus according to claim 30, wherein the optimization unit generates a pair instruction for transferring two or more kinds of data at the same time regarding a memory access instruction for accessing the data to be allocated in the memory region (see for example, p.181, lines 26-29, "use the ldd and std (doubleword load and store) instructions")

Claim 31:

Stallman further discloses the compiler apparatus according to claim 29,

- wherein the directive acquisition unit acquires a directive for alignment of structure data together with a directive for translating the source program (see for example, p.10, line 3, "-O –O0 –O1 –O2 -O3 –Os" and related text), and
- the optimization unit allocates the structure data declared in the source program in the memory region so that its head address matches the alignment (see for example, p.177, section 5.33 Specifying Attributes of Variables. "aligned (alignment)" and related description, also see example,

p.178, line 1, int x __attribute__ ((aligned (16))) = 0 ":" and related description).

Claim 41:

Stallman also discloses the compiler apparatus according to claim 31, wherein the optimization unit generates a pair instruction for transferring two or more kinds of data at the same time regarding a memory access instruction for accessing the data to be allocated in the memory region (see for example, p.181, lines 26-29, "use the ldd and std (doubleword load and store) instructions").

Claim 32:

Stallman further discloses the compiler apparatus according to claim 29,

- wherein the directive acquisition unit detects designation of alignment of data that a pointer variable of argument shown by the name of a specific variable indicates in the source program (see for example, p.182, lines 16-20, "If you declare or use arrays of variable of an efficiently-aligned type...") and
- the optimization unit performs the optimization assuming that the data that is an object of designation detected by the directive acquisition unit is allocated in the memory region by the designated alignment (see for example, p.182, lines 16-20, "the compiler generates for these pointer arithmetic operations...").

Claim 42:

Stallman also discloses the compiler apparatus according to claim 32, wherein the optimization unit generates a pair instruction for transferring two or more kinds of data at the same time regarding a memory access instruction for accessing the data to be allocated in the memory region (see for example, p.181, lines 26-29, "use the ldd and std (doubleword load and store) instructions").

Claim 33:

Stallman further discloses the compiler apparatus according to claim 29,

- wherein the directive acquisition unit detects designation of alignment of data that a local pointer variable shown by the name of a specific variable indicates in the source program (see for example, p.182, lines 16-20, "If you declare or use arrays of variable of an efficiently-aligned type...") and
- the optimization unit performs the optimization assuming that the data that is an object of designation detected by the directive acquisition unit is allocated in the memory region by the designated alignment (see for example, p.182, lines 16-20, "the compiler generates for these pointer arithmetic operations...").

Claim 43:

Stallman also discloses the compiler apparatus according to claim 33, wherein the optimization unit generates a pair instruction for transferring two or more kinds of data at the same time regarding a memory access instruction for accessing the data to be allocated in the memory region (see for example, p.181, lines 26-29, "use the ldd and std (doubleword load and store) instructions").

Claim Rejections - 35 USC § 103

15. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

16. Claims 35 and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stallman (Richard M. Stallman, Using and Porting the GNU Compiler Collection for GCC 3.1)

Claim 35:

Claim 35 claims a computer source program/code that is written according to the requirement of compiler apparatus discussed in claims 29-34 above. Because all claimed limitations of said compiler apparatus have been address and/or set forth above in claims 29-34. Therefore, it would have been obvious to perform the

feature that the directive specified while using such compiler apparatus to compile said computer source code.

Claim 40:

Claim 40 is a software program product (GNU gcc Compiler) version of claimed apparatus in claims 29-34 and 41-43 above, wherein all claimed limitations have been address and/or set forth above. Therefore, as the references teach all the limitation of claims 29-34 and 41-43, they also teach the limitations of claim 40. Thus, it also would have been obvious.

17. Claims 18-23 and 38 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stallman (Richard M. Stallman, Using and Porting the GNU Compiler Collection for GCC 3.1) in view of Geva (Robert Y. Geva, US 6,539,541)

Claim 18:

Stallman discloses the compiler apparatus according to claim 14 and also discloses the compilation option "-funroll-loops" can only be done for loops whose number of iterations can be determined at compile time or run time (see for example, p.55, lines 38-42, "This is only done for loops whose number of iterations can be determined at compile time or run time),.

But does not disclose detecting a directive of designation of the number of iteration. However, Geva in the same analogous art of loop unrolling discloses

the number of iterations defined by the directive (see for example, col.9, lines 37-38, "where the value of loop iterations is read by the program from an input file"). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to specify the number of iterations of specific loop processing in the source program. One would have been motivated to do so to ensure the number of iterations can be determined at compile time and further guarantee the loop unrolling can be performed as suggest by Geva (see for example, col.9, lines 31-42).

Claim 19:

Stallman and Geva disclose the compiler apparatus according to claim 18, Geva further discloses: the optimization unit restrains generation of an escape code that is needed in the case of the number of the iterations being 0 when the minimum number is 1 or more (see for example, col.10, lines 9-10, "When the unrolled loop is a counted loop, there is no need to test for the exit condition inside the unrolled body.").

Claim 20:

Stallman and Geva disclose the compiler apparatus according to claim 18, Geva further disclose the optimization unit performs the optimization by loop unrolling when the minimum number is equivalent to or more than the number of development by the loop unrolling (see for example, col.10, lines 6-9, "One such

optimization may be loop unrolling where a loop is unrolled 'n' times, such that 'n-1' additional copies of the loop body are made").

Claims 21 and 22:

Stallman and Geva disclose the compiler apparatus according to claim 18, neither of them explicitly disclose the number of iteration is even/odd number. However, Geva in the same analogous art of loop unrolling discloses the number of iterations defined by the directive (see for example, col.9, lines 37-38, "where the value of loop iterations is read by the program from an input file"). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to specify the number of iterations of specific loop processing in the source program. One would have been motivated to do so to ensure the number (even/odd number) of iterations can be determined at compile time and further guarantee the loop unrolling can be performed as suggest by Geva (see for example, col.9, lines 31-42).

Claim 23:

Claim 13 claims a computer source code that is written according to requirement of compiler apparatus discussed in claims 14-18 above. Because all claimed limitations of said compiler apparatus have been address and/or set forth above in claims 14-18. Therefore, it would have been obvious to perform the feature

that the directive specified while using such compiler apparatus to compile said computer source code.

Claim 38:

Claim 38 is a software program product version of claimed apparatus in claims 14-22 above, wherein all claimed limitations have been address and/or set forth above. Therefore, as the references teach all the limitation of claims 14-22, they also teach the limitations of claim 38. Thus, it also would have been obvious.

18. Claims 1-5 and 36 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stallman (Richard M. Stallman, Using and Porting the GNU Compiler Collection for GCC 3.1) in view of Nakamura (Nakamura et al., Architecture and Compiler Co-Optimization for High Performance Computing)

Claim 1:

Stallman discloses a compiler apparatus that translates a source program into a machine language program comprising:

- a directive acquisition unit operable to acquire a directive for optimizing a machine language program to be generated (see for example, p.9, "Optimization options") and
- an optimization unit operable to perform optimization by generating a sequence of machine language instructions following an acquired directive

(see for example, see p.49, section 3.10, "Options that Control Optimization"),

But does not explicitly disclose, wherein the optimization unit performs the optimization by deciding array data allocated to a global memory region following a directive when the directive acquisition unit acquires the directive on the array data to be allocated to the global memory region. However, Nakamura in the same analogous art of compiler optimization discloses using directive to allocate array in software controllable memory (see for example, p.52, section 3, Directive-Based Compiler, also see p.53, section 3.2 Example of Directives). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to add this specific optimization option to Stallman's gcc compiler. One would have been motivated to do so to improve the performance for high performance computing as suggested by Nakamura (see for example, p.51, section 2.2 Benefit of SCM)

Claim 2:

Stallman and Nakamura disclose the compiler apparatus according to claim 1, Nakamura further discloses:

- wherein the directive acquisition unit acquires designation of a maximum data size of array data to be allocated to a global memory region together with a directive for translating the source program (see for example, p.52,

Figure 2. directive "!$scm begin (<array_name>...)" and related text, also see, p.53, section Example of Directives"),

But neither of them discloses

- the optimization unit, out of array data declared by the source program, allocates array data whose maximum data size does not exceed the maximum data size to a global memory region and array data whose maximum data size exceeds the maximum data size to a memory region out of the global memory region.

However, it is well known in the computer art that software compiler allocates array variable automatically in the certain location of memory according to source code declaration. Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use directive to specify the specific memory location where the compiler allocates the array variable to according to the predefined parameter (maximum data size). One would have been motivated to do so to improve the performance for high performance computing as suggested by Nakamura (see for example, p.51, section 2.2 Benefit of SCM)

Claims 3-4:

Stallman and Nakamura discloses the compiler apparatus according to claim 1, Nakamura further disclose:

- wherein the directive acquisition unit detects a directive for not allocating/ allocating specific array data to the global memory region in the source program (see for example, p.52, Figure 2. directive "!$scm begin (<array_name>...)" and related text, also see, p.53, section Example of Directives"), and
- the optimization unit allocates array data that are an object of a directive detected by the directive acquisition unit to a memory region out of the global memory region/global memory region (see for example, p.53, section Example of Directives")

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to add this specific optimization option to Stallman's gcc compiler. One would have been motivated to do so to improve the performance for high performance computing as suggested by Nakamura (see for example, p.51, section 2.2 Benefit of SCM)

Claim 5:

Stallman discloses a computer-readable recording medium on which a source program described in a high-level language is recorded, wherein the source program includes at least one of descriptions for directing a compiler that translates the source program into a machine language program.

But does not discloses, (1) not to allocate a specific array data to a global memory region and (2) to allocate the specific array data to the global memory region. However, Nakamura in the same analogous art of compiler optimization discloses using directive to allocate array in software controllable memory (see for example, p.52, section 3, Directive-Based Compiler, also see p.53, section 3.2 Example of Directives). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to add this specific optimization option to Stallman's gcc compiler. One would have been motivated to do so to improve the performance for high performance computing as suggested by Nakamura (see for example, p.51, section 2.2 Benefit of SCM)

Claim 36:

Claim 36 is a software program product version of claimed apparatus discussed in claims 1-4 above, wherein all claimed limitations have been address and/or set forth above. Therefore, as the references teach all the limitation of claims 1-4, they also teach the limitations of claim 36. Thus, it also would have been obvious.

19. Claims 6-8 and 13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stallman (Richard M. Stallman, Using and Porting the GNU Compiler

Collection for GCC 3.1) in view of PPCREF (PPCREF Project – High level design, Software Pipelining and Branch Optimization)

Claim 6:

Stallman discloses a compiler apparatus that translates a source program into a machine language program comprising:

- a directive acquisition unit operable to acquire a directive for optimizing a machine language program to be generated (see for example, p.9, "Optimization options"); and
- an optimization unit operable to perform optimization by generating a sequence of machine language instructions following an acquired directive (see for example, see p.49, section 3.10, "Options that Control Optimization"),

But Stallman does not explicitly disclose using directive to perform optimization. However, it is well known in the computer art that gcc compiler does support software pipelining at back end (see for example, gcc's source code "reorg.c" implement software pipelining by delay slot scheduling). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to add directive to turn on/off software pipelining optimization feature while compiling. One would have been motivated to do so to compile program code more flexible.

Claim 7:

Stallman and PPCREF disclose the compiler apparatus according to claim 6, Stallman further discloses: some other directives can be used to optimize compilation for the source program (see for example, p.10, line 3, "-O –O0 –O1 –O2 -O3 –Os" and related text).

But neither of them discloses using directive to turn off software pipelining optimization feature of the gcc compiler for compile whole software program. However, Stallman also disclose two forms of compilation option (see for example, p.7, section 3, GCC command options, lines 22-23, "positive and negative forms"(perform/not performing)). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use this negative form option to direct compiler not performing software pipelining. One would have been motivated to do so to compile software program more flexible.

Claim 8:

Stallman and PPCREF disclose the compiler apparatus according to claim 6, Stallman also disclose a directive format in gcc (see for example, p.25, lines 19-22, "#pragma implementation" and also see p.24, lines 30-35, "#pragma interface". but neither of them explicitly discloses

- wherein the directive acquisition unit detects a directive for not performing the optimization by software pipelining of a specific loop processing in the source program, and

- the optimization unit restrains the optimization by software pipelining of loop processing that is an object of the directive detected by the directive acquisition unit.

However, it would have been obvious to one having ordinary skill in the art at the time the invention was made to implement similar "#pragma" directive to direct compiler not performing software pipelining of a specific loop processing by inserting the directive into the program source code. One would have been motivated to compile software program more flexible.

Claim 13:

Claim 13 claims a computer source code that is written according to requirement of compiler apparatus discussed in claims 6-10 above. Because all claimed limitations of said compiler apparatus have been address and/or set forth above in claims 6-10. Therefore, it would have been obvious to perform the feature that the directive specified while using such compiler apparatus to compile said computer source code.

20. Claims 9-12 and 37 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stallman (Richard M. Stallman, Using and Porting the GNU Compiler Collection for GCC 3.1) in view of PPCREF (PPCREF Project – High level

design, Software Pipelining and Branch Optimization) in further view of Granston (US 6,892,380)

Claims 9-10:

Stallman and PPCREF disclose the compiler apparatus according to claim 6, PPCREF further discloses insert prolog and epilog portion to increase execution but neither of them discloses:

- wherein the directive acquisition unit detects a directive for performing the optimization by software pipelining that removes/does not remove a prolog portion and an epilog portion of a specific loop processing in the source program, and
- the optimization unit performs the optimization by software pipelining of loop processing that is an object of the directive detected by the directive acquisition unit whenever possible to remove the prolog portion and the epilog portion.

However, Granston in the same analogous art of software pipelining discloses a method to remove (omit) the epilog to reduce code size (see for example, p.4, lines 19-26, also see example at p.2 and related text). It is also well known in the computer art that the compilation directive can be added to selectively optimize the specific loop. One would have been motivated to compile software program more flexible.

Claim 11:

Stallman and PPCREF disclose the compiler apparatus according to claim 6, but neither of them discloses:

- wherein the directive acquisition unit detects designation of the number of iterations of specific loop processing in the source program, and
- the optimization unit performs optimization of loop processing that is an object of the designation detected by the directive acquisition unit based on the designated number of iterations.

However, Granston in the same analogous art of software pipelining discloses a designation of the number of iterations of specific loop (see for example, col.1, lines 29-39, "n represents the number of desired iterations"). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to specifically define the number of iteration. One would have been motivated to do so to use this number compare with minimum required trip count to decide executing pipelined version or original version as indicated by Granston (see for example, col.3, lines 43-49, "if (n>= min required trip count")

Claim 12:

Stallman, PPCREF and Granston disclose the compiler apparatus according to claim 11, Granston further discloses:

- wherein the designation of the number of the iterations is the minimum number by which the loop processing is iterated (see for example, col.3, lines 43-49, "if (n>= min required trip count"), and

- the optimization unit performs the optimization by software pipelining when the minimum number is equivalent to or larger than the number of iterations that overlap by software pipelining (see for example, col.3, lines 43-49, "if (n>= min required trip count; pipelined version; else original version; endif" and relate description).

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to specifically define the number of iteration as the minimum iterated number. One would have been motivated to do so to use this number to generate multiversion code indicated by Granston (see for example, col.3, lines 50-51, "This technique is referred to as multiversion code generation)

Claim 37:

Claim 37 is a software program product version of claimed apparatus in claims 6-12 above, wherein all claimed limitations have been address and/or set forth above. Therefore, as the references teach all the limitation of claims 6-12, they also teach the limitations of claim 37. Thus, it also would have been obvious.

21. Claims 24-28 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stallman (Richard M. Stallman, Using and Porting the GNU Compiler Collection for GCC 3.1) in view of Prata (Stephen Prata, C Primer Plus, Third Edition)

Claim 24:

Stallman discloses a compiler apparatus that translates a source program into a machine language program comprising:

- a directive acquisition unit operable to acquire a directive for optimizing a machine language program to be generated (see for example, p.9, "Optimization options"); and
- an optimization unit operable to perform optimization by generating a sequence of machine language instructions following the acquired directive (see for example, see p.49, section 3.10, "Options that Control Optimization"),

but does not explicitly disclose, wherein the optimization unit performs optimization on an "if" conversion following a directive when the directive acquisition unit acquires the directive on the "if" conversion. However, Prata in the same analogous art of conditional compilation discloses using the #ifdef, #else, and #endif directive to set up conditional compilation (see for example, p.576, Other Directives and p.577-581, Conditional Compilation). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use #define, #ifdef, #else and #endif with the "if" conversion. One would have been motivated to do so to selectively compile

software program as taught by Prata (see for example, p.577, section "Conditional Compilation", "You can use them to tell the compiler to accept or ignore blocks of information or code according to conditions at the time of compilation.")

Claims 25-27:

Stallman and Prata disclose the compiler apparatus according to claim 24, Stallman further discloses:

- a directive for translating the source program (see for example, p.10, line 3, "-O –O0 –O1 –O2 -O3 –Os" and related text)

Prata also discloses:

- wherein the directive acquisition unit detects a directive for making/not making an "if" conversion of a specific "if" structure sentence in the source program (see for example, p.577-581, Conditional Compilation), and
- the optimization unit restrains/makes the "if" conversion of all "if" structure sentences in the source program when the directive acquisition unit acquires the directive (#define, #ifdef, #endif) for not making/making the "if" conversion (see for example, p.577-581, Conditional Compilation)

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to combine Stallman's compilation directive option with Prata's conditional compilation directive or directly use Stallman's gcc compiler to compile source code containing Prata's conditional compilation

directive to make/not make "if" conversion as taught by Prata (see for example, p.577, section "Conditional Compilation", "You can use them to tell the compiler to accept or ignore blocks of information or code according to conditions at the time of compilation.")

Claim 28:

Claim 28 claims a computer source code that is written according to requirement of compiler apparatus discussed in claims 24-27 above. Because all claimed limitations of said compiler apparatus have been address and/or set forth above in claims 24-27. Therefore, it would have been obvious to perform the feature that the directive specified while using such compiler apparatus to compile said computer source code.

Claim 39:

Claim 39 is a software program product version of claimed apparatus in claims 24-27 above, wherein all claimed limitations have been address and/or set forth above. Therefore, as the references teach all the limitation of claims 24-27, they also teach the limitations of claim 39. Thus, it also would have been obvious.

Conclusion

22. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Zheng Wei whose telephone number is (571) 270-1059. The examiner can normally be reached on Monday-Thursday 8:00-15:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature of relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571- 272-1000.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

ZW

TUAN DAM
SUPERVISORY PATENT EXAMINER